DIFFERENTIAION

Ari Maller

DERIVATIVES

- Derivatives are as important in physics as integrals, but you will hear much less about numerical derivatives than integrals for a number of reasons.
  - The basic technique for numerical differentiation is quite simple.
  - Derivatives of known functions can always be calculated analytically, so there is less need to use numerical methods.
  - There are some significant practical problems with numerical derivatives, so they are used less often, that is avoided if possible.

FORWARD AND BACKWARD DIFFERENCES

- The basic method for calculating derivatives is the straightforward approximation of the standard derivative formula.

$$\frac{df}{dx} = \frac{f(x+h) - f(x)}{h}$$

- This is called the *forward difference* since x+h is greater than x. Alternatively we can evaluate the *backward difference*

$$\frac{df}{dx} = \frac{f(x) - f(x-h)}{h}$$

- Either formula should give one the same result, though sometimes one is preferred like if there is a discontinuity at x.

ERRORS

- In order to decide how to choose h we will need to understand the errors in our evaluation.
- For derivatives rounding error and approximation error are likely to both contribute to our error. To determine the approximation error let us Taylor expand around f(x+h)

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + O(h^3)$$

- To leading order the error on the forward difference is 1/2h|f''(x)|, because we divide everything by h to define a derivative. There is also a rounding error that is much more important than in an integral because we are performing subtraction. The rounding error should be 2C|f(x)|/h.

ERROR

- So the total error will be

$$\epsilon = \frac{2C|f(x)|}{h} + \frac{1}{2} h f''(x)$$

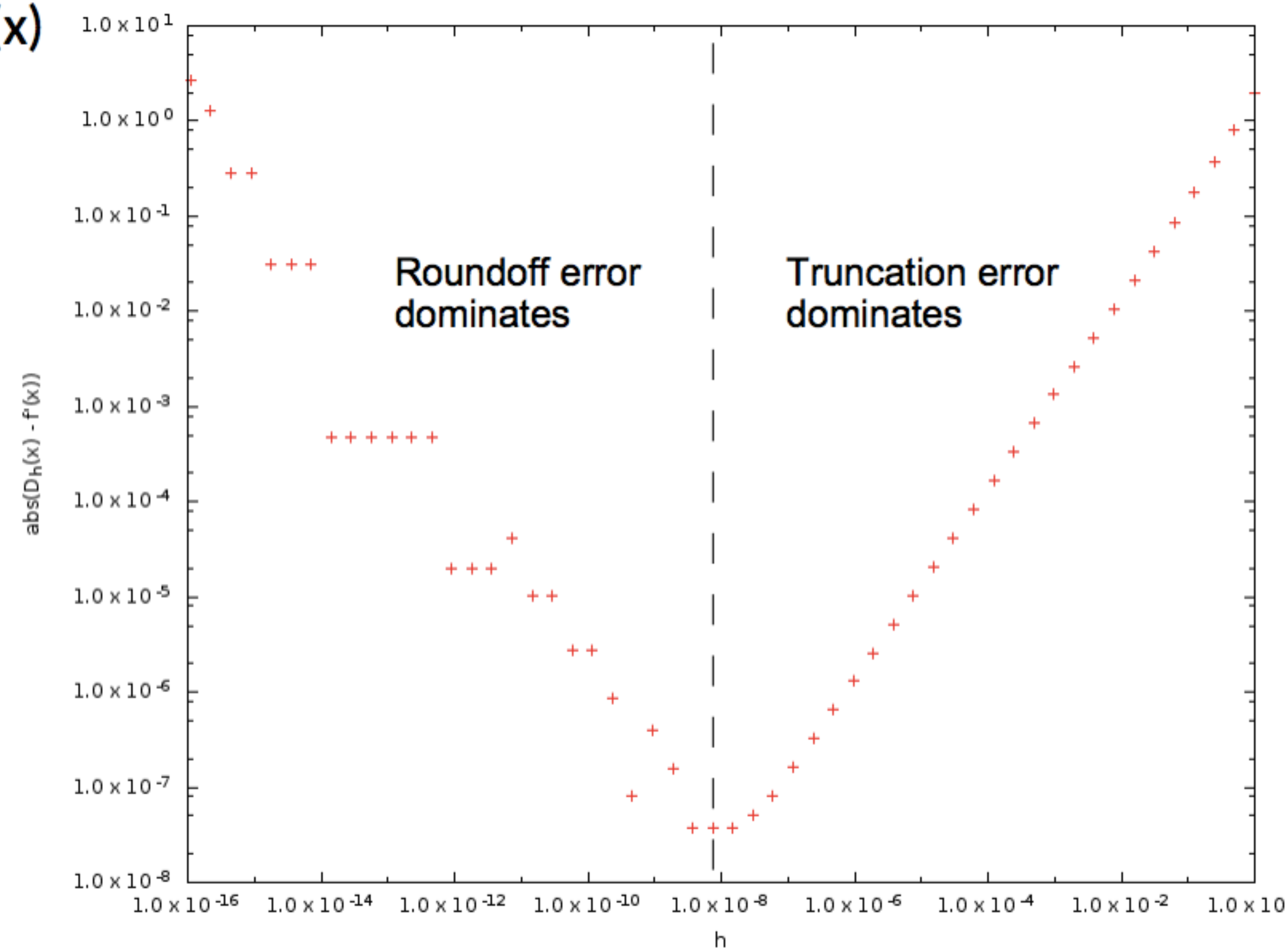
- If we try to minimize this equation by taking the derivative with respect to h and setting the derivative to zero we get

$$-\frac{2C|f(x)|}{h^2} + \frac{1}{2}|f''(x)| = 0 \qquad => h = \sqrt{4C|\frac{f(x)}{f''(x)}|}$$

- so if f(x) and f''(x) are of order unity then we should take h$\sim 10^{-8}$ in Python and the error in our derivative would also be about 10^{-8}. This is much worse then we typically were able to do with integration where we could get to machine precision in a reasonable number of steps.

ERROR

exp(x)



1.0 x 10^1
1.0 x 10^0
1.0 x 10^-1
1.0 x 10^-2
1.0 x 10^-3
1.0 x 10^-4
1.0 x 10^-5
1.0 x 10^-6
1.0 x 10^-7
1.0 x 10^-8
abs(D_h(x) - f'(x))
Roundoff error dominates
Truncation error dominates
1.0 x 10^-16
1.0 x 10^-14
1.0 x 10^-12
1.0 x 10^-10
1.0 x 10^-8
1.0 x 10^-6
1.0 x 10^-4
1.0 x 10^-2
1.0 x 10^0
h

CENTRAL DIFFERENCES

- One way to improve the accuracy of our derivative is to combine the forward and backward differences into a central difference

$$\frac{df}{dx} = \frac{f(x+h/2) - f(x-h/2)}{h}$$

- Taylor expanding around these two points will give alternating signs for odd terms cancelling them out. So our error will become

$$\epsilon = \frac{2C|f(x)|}{h} + \frac{1}{24}h^2|f'''(x)| \qquad => h = (24C|\frac{f(x)}{f'''(x)}|)^{1/3}$$

- So now for Python h should be 10^{-5} and the error will be 10^{-10}, a hundred fold improvement from the forward or backward differences. Notice h actually gets bigger while the error gets smaller.

SAMPLED FUNCTIONS

- If we have sampled data spaced h apart like from an experiment then the central difference at a point would need to be calculated x+h and x-h apart instead of x+h/2 and x-h/2.
- However, we could achieve better accuracy if instead we calculated the derivative for points in between our sampled points.
- Thus in situations where we know f(x) for certain points x_k, it is better to evaluate the derivative for points in-between x_k and not at x_k.

EXERCISE 5.15

- Create a user-defined function f(x) that returns the value 1 + 1/2tanh(2x).
- Then use a central difference to calculate the derivative of the function in the range [-2,2].
- The derivative of this function is $\text{sech}^2(2x)$. Make a graph with your numerical result and the analytic answer on the same plot. It may help to plot the exact answer as lines and the numerical one as dots.
- (Hint: In Python the tanh function is found in the math package, and it's called simply tanh.)

$$\frac{df}{dx} = \frac{f(x+h/2) - f(x-h/2)}{h}$$

HIGHER ORDER DERIVATIVES

- The forward and backward difference are basically a linear fit to two points and then taking the slope of that as the derivative.
- Like in the case of integrals, can we try higher order fits?
- If we try a quadratic fit we will get central difference.
- A fourth order fit would be

$$f'(x) \simeq \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}$$

- Just like for integrals at higher order you use more points and they are weighed differently.

2ND DERIVATES

- We can just as easily compute 2nd derivates by simply recognizing that if g(x) = f'(x), then f''(x) = g'(x). If we used the central difference for both calculations we would get

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

- The error on the second derivative goes as

$$\epsilon = \frac{4C|f(x)|}{h^2} + \frac{1}{12}h^2|f'''(x)|$$

- In this case we see that we want h$\sim 10^{-8}$ and the error will be $\sim 10^{-8}$, the same as for using forward or backward difference.
- So taking a second derivative has basically reduced our accuracy the same as using a method of one lower order.

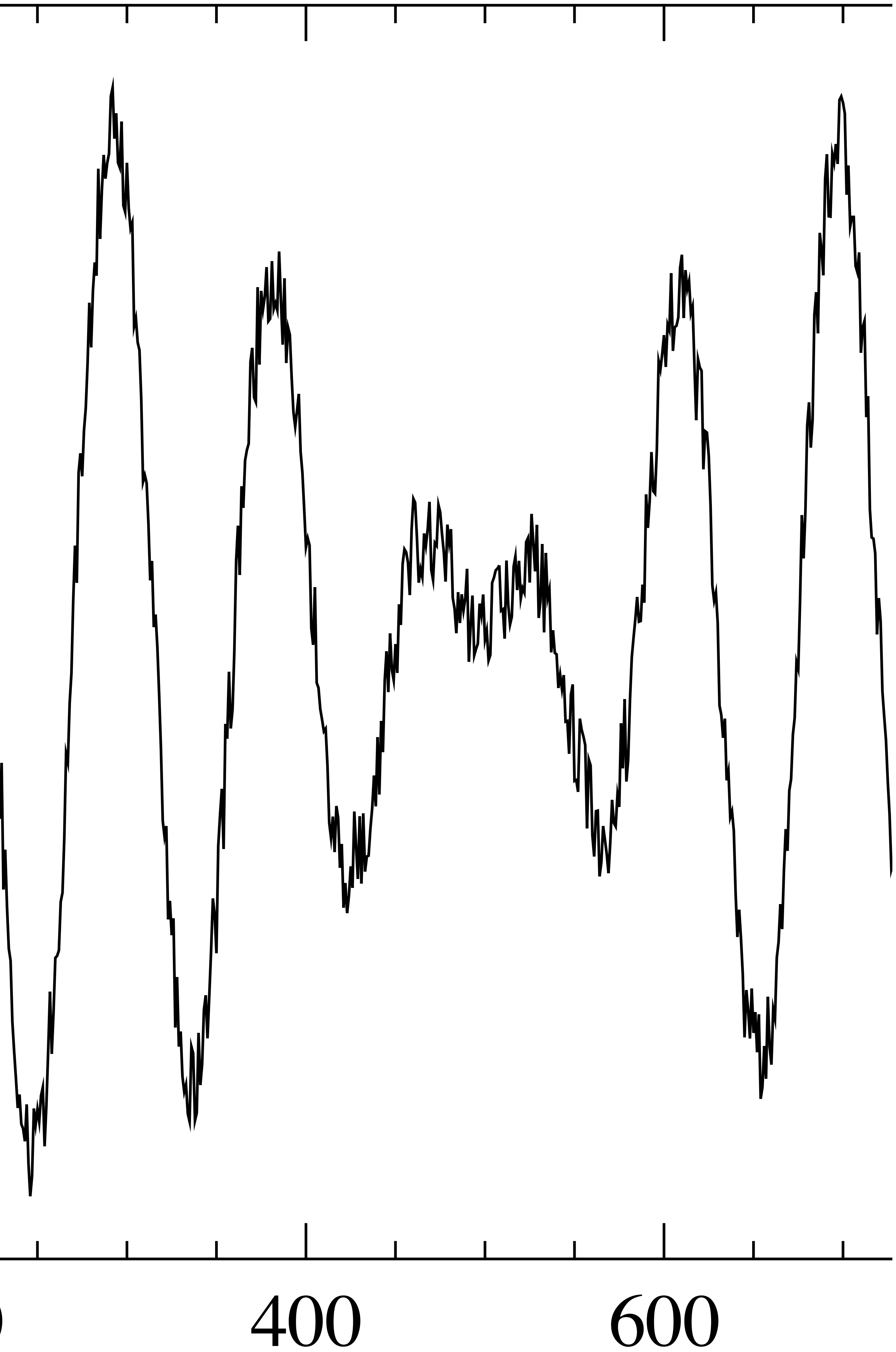
PARTIAL DERIVATIVES

- We can determine partial derivatives just as easily using the central difference method or any other method. We simply hold all variables fixed except the one that we are taking the derivative off.

$$\frac{\partial f}{\partial x} = \frac{f(x+h/2,y) - f(x-h/2,y)}{h}$$

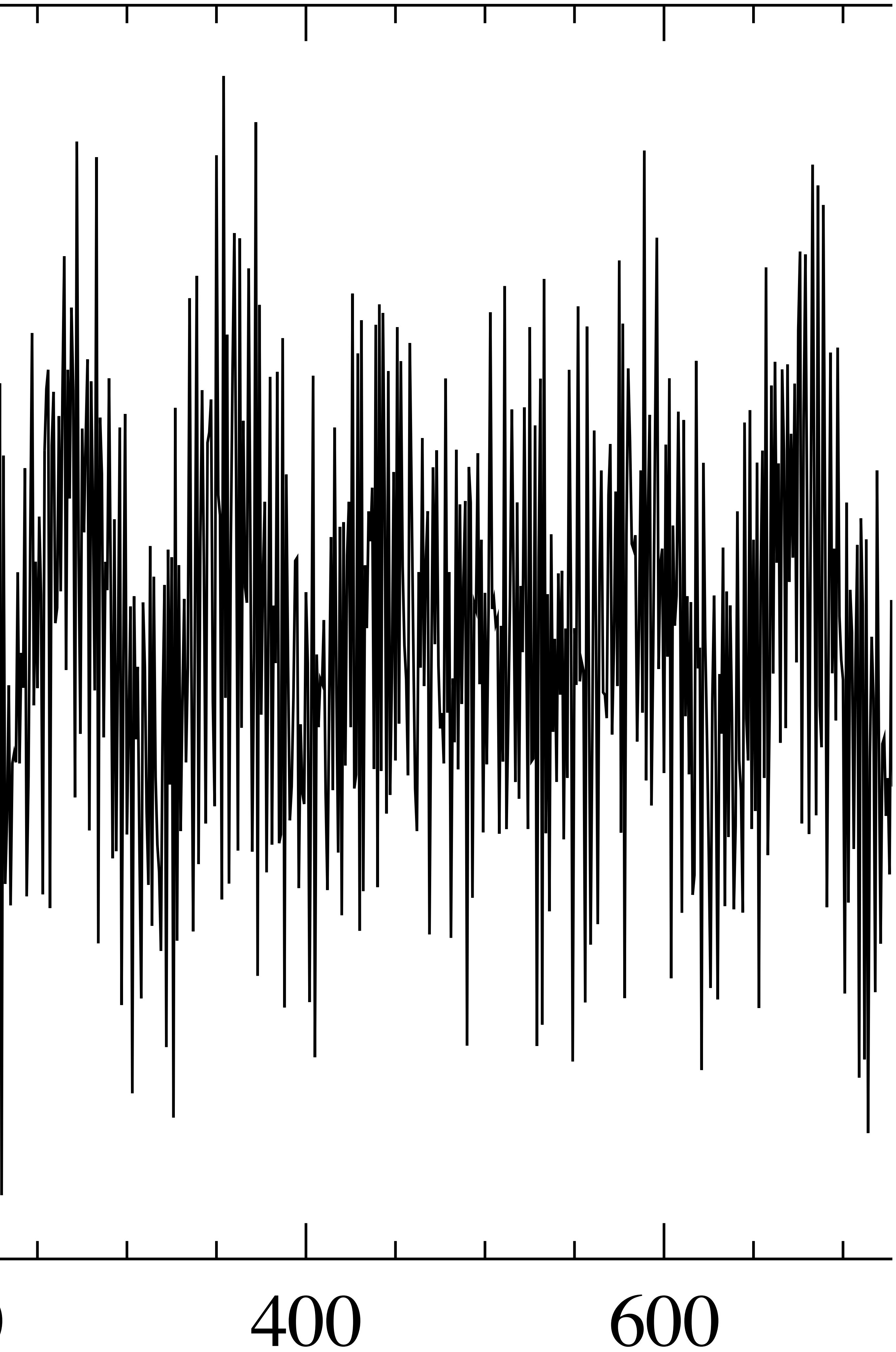
$$\frac{\partial f}{\partial y} = \frac{f(x,y+h/2) - f(x,y-h/2)}{h}$$

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{f(x+h/2,y+h/2) - f(x-h/s,y+h/2) - f(x+h/2,y-h/2) + f(x-h/2,y-h/2)}{h}$$



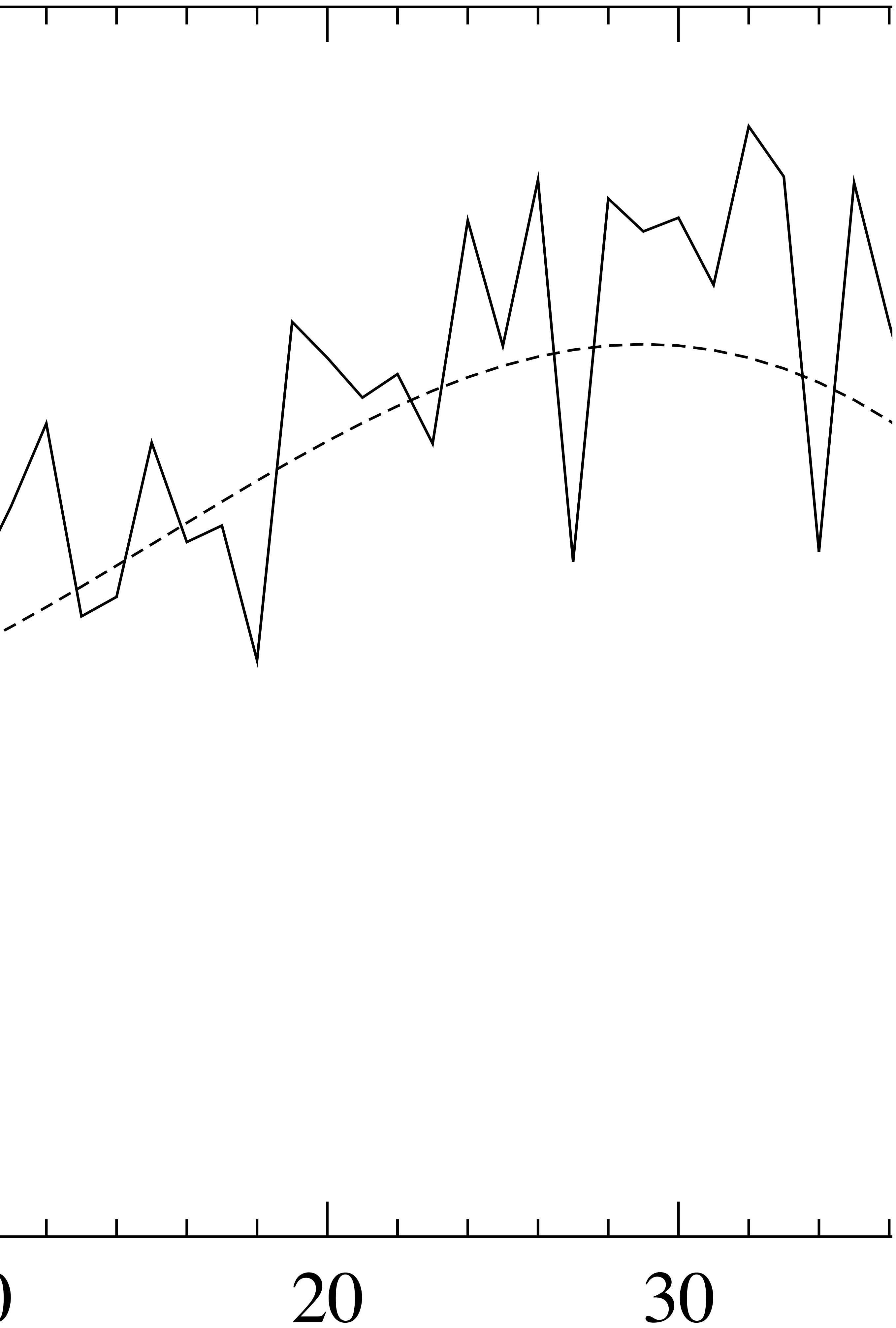
NOISY DATA

- One tricky thing with derivatives can be if you have noisy data.
- Taking the points as is the curve to the left will produce derivatives with large variance.
- If we think the variation is real then that is what we want. But if we think the variation is just noise and we want the derivative of some underlying function, then we don't want the simple derivative of the recorded values.



NOISY DATA

- The derivative taken just with the measured points is this plot.
- If we know this is due to noise there are a few things we can do.
  - The easiest is to just make h larger. We can treat the noise like rounding error and find a value for h that minimizes the contribution of the noise.



NOISY DATA

- A second option is to fit a curve to the portion of the data where we wish to take the derivative. This is not a fit to just a few points like a higher order derivative, but fit to scales large enough to see the underlying function and not the noise.
- A third option is to smooth the data before taking the derivative. This could be done with a Fourier transform for example.

NUMPY.GRADIENT AND SCIPY.MISC.DERIVATIVE

- The function derivative() can be found in scipy.misc and will calculate the nth derivative of a given function using central difference.
- The call is derivative(func, x0, dx=0.1, n=1)
- Alternatively one can use numpy's gradient() function to calculate derivates from an array of values.
- The call is gradient(y,axis=0). If no axis is given then the derivatives are calculated along all axes, which give you gradients.

TERMINOLOGY

- Forward or Backward difference - a numerical derivative where the difference is taken as f(x+h) - f(x) or f(x) - f(x-h).
- Central difference - a numerical derivative where the difference is taken between f(x+h/2) - f(x-h/2).